1. En attendant...

Comment installer le logiciel oXygen XML editor !

- visitez https://www.oxygenxml.com/download.html
- séléctionnez XML Editor et cliquez Download
- séléctionnez votre plateforme (Windows, Mac, Linux...) et cliquez Download
- completez le formulaire en fournissant votre adresse email, votre pays, votre désir de pub, et le code secret en bas
- enregistrez l'installeur et l'activez
- controllez votre email pour le licence d'essai: vous en aurez besoin la première fois que vous activez oXygen

2. Pour commencer: un premier exemple simplissime

Nous pouvons utiliser ODD pour definir n'importe quel systeme de balisage.

Imaginons un projet qui souhaite baliser dans ses documents des éléments `<book>`, qui contiennent un mélange d'éléments `<para>`s et `<image>`s. Ce projet ne connait rien de la TEI, et n'en a pas envie. De même pour les espaces de noms.

Voici une esquisse de la documentation ODD d'un tel projet. La partie essentielle sera le *schemaSpec* (specification de schema) qui contiendra trois *elementSpec* (specifications d'element)...

```
<schemaSpec ns="" start="book"
 ident="bookSchema">
 <elementSpec ident="book">
  <desc>Élément racine d'un schéma simplissime pour encoder les livres</desc>
  <content>
   <alternate maxOccurs="unbounded">
    <elementRef key="para"/>
    <elementRef key="image"/>
   </alternate>
  </content>
 </elementSpec>
 <elementSpec ident="para">
  <desc>un paragraphe de text </desc>
  <content>
   <textNode/>
  </content>
 </elementSpec>
 <elementSpec ident="image">
  <desc>un élément vide qui pointe sur un fichier graphique</desc>
  <content/>
  <attList>
   <attDef ident="href">
    <desc>fournit l'URI de l'objet ciblé</desc>
    <datatype>
     <dataRef name="anyURI"/>
    </datatype>
   </attDef>
  </attList>
 </elementSpec>
</schemaSpec>
```

3. So what ?

- On peut générer un schéma RELAX NG, W3C, ou DTD à l'aide d'une transformation XSLT
- On peut extraire du meme document les fragments documentaires, notamment les descriptions des éléments et des attributs

TEI fournit un élément spécialisé pour cela :

```
<specList>
 <specDesc key="para"/>
 <specDesc key="picture"/>
</specList>
```

Ce balisage généra quelque chose comme

```
<list>
 <item>
  <label>
   <gi>para</gi>
  </label> : un paragraphe de texte </item>
 <item>
  <label>
   <gi>picture</gi>
  </label> : un élément vide qui pointe sur un fichier
```

```
   graphique</item>
</list>
```

4. Essayons cela avec oXygen…

- Démarrez oXygen
- Créez un nouveau fichier (CTRL-N)
- Sélectionnez TEI-P5 -> ODD Customization dans le menu Cadre des modèles du dialog Nouveau
- Remplacer l'élément <schemaSpec> (et ses balises et son contenu) proposé par le contenu du fichier oddex-1.xml
- Insérez le contenu du fichier oddex-1-doc.xml *avant* le nouveau <schemaSpec>
- Enregistrez votre ODD sous le nom oddex-1.odd
- Sélectionnez les Scénario de Transformation TEI ODD to RELAX NG Compact et TEI ODD to HTML pour générer un schéma et sa documentation à partir de cet ODD
- Lire la documentation
- Validez le fichier test oddex-1-test.xml contre le schéma out/oddex-1.rnc que vous venez de générer: Ou créez un nouveau document XML conforme à ce schéma et controlez les balises proposées par oXygen.

5. Notions de classe (1)

Dans le monde réel, le contenu de nos <book> ne se limite pas uniquement aux <para>s et aux <image>s. On peut regrouper tous les éléments qui peuvent apparaître au sein d'un book : nous appelons ce regroupement une *classe*, pour laquelle nous proposons le nom bookPart.

Voici la définition de la classe bookPart :

```
<classSpec ident="bookPart" type="model">
 <desc>éléments qui ont la possibilité de figurer dans un <gi>book</gi>
 </desc>
</classSpec>
```

Pour indiquer son adhésion à une classe, la spécification d'un élément utilise l'élément <classes> :

```
<elementSpec ident="para">
 <desc>un paragraphe de text </desc>
 <classes>
  <memberOf key="bookPart"/>
 </classes>
 <content>
  <textNode/>
 </content>
</elementSpec>
```

Maintenant, au lieu de lister exhaustivement tous les composants possibles d'un <book>, il suffit de dire que cet élément est composé des membres de la classe bookPart .

```
<elementSpec ident="book">
 <desc>Élément racine d'un schéma simplissime pour encoder les livres</desc>
 <content>
  <classRef key="bookPart" minOccurs="1"
   maxOccurs="unbounded"/>
 </content>
</elementSpec>
```

(Dès que nous découvrirons l'existence de listes dans les livres nous saurons quoi faire)

6. Définition d'une classe d'attribut

Dans le monde réel, il est très probable que plusieurs éléments différents comportent les même attributs : il sera donc très pratique de les définir en une seule fois

ODD nous permet de dire que tous les éléments ayant en commun un ensemble d'attributs constituent une *classe d'attribut*; nous la définissons ainsi:

```
<classSpec ident="pointing" type="atts">
 <desc>regroupe les éléments comportant l'attribut <att>href</att>
 </desc>
 <attList>
  <attDef ident="href">
   <desc>fournit l'URI de l'objet ciblé</desc>
   <datatype>
    <dataRef name="anyURI"/>
   </datatype>
  </attDef>
 </attList>
</classSpec>
```

Les attributs d'un élément sont ainsi spécifiables :

- directement avec un <attList> dans son <elementSpec>
- indirectement par l'adhésion de l'élément à une classe d'attribut

7. Testez votre compréhension

- Ouvrez le fichier oddex-2.odd avec oXygen et comparez le avec oddex-1.odd
- Créez une nouvelle version du schéma à partir de cet ODD
- Assurez vous que le fichier test oddex-1-test.xml reste valide contre cette nouvelle version du schéma
- Comprenez-vous l'effet des références aux classes?

8. Contrôle des valeurs d'un attribut

Les valeurs possibles d'un attribut peuvent être contrôlées de plusieurs manières :

- Par référence à un *datatype* (type de donnée) externe, par ex anyURI ou ID ou numeric(ce sont des datatypes standard, définis par le W3C)
- En fournissant notre propre liste des valeurs avec l'élément <valList>
- Par référence à un *datatype* (type de donnée) interne défini par un <dataSpec> interne à notre ODD

Par exemple:

```
<classSpec ident="bookAtts" type="atts">
 <desc>attributs applicables aux <gi>book</gi>
 </desc>
 <attList>
  <attDef ident="xml:id">
   <desc>fournit un identifiant unique pour le livre</desc>
   <datatype>
    <dataRef name="ID"/>
   </datatype>
  </attDef>
  <attDef ident="status">
   <desc>indique le statut du livre </desc>
   <valList>
    <valItem ident="red"/>
    <valItem ident="green"/>
    <valItem ident="unknown"/>
   </valList>
  </attDef>
  <attDef ident="cote">
   <desc>indique la cote ou se trouve le livre (un numéro entre 1 et 9999)</desc>
   <datatype>
    <dataRef key="myShelfNo"/>
   </datatype>
  </attDef>
 </attList>
</classSpec>
```

(Il faut aussi fournir un <dataSpec> pour definir myShelfNo)

9. Tester votre compréhension…

- Insérez dans votre fichier oddex-2.odd le fichier oddex-2x.xml: votre ODD inclut maintenant la declaration d'une nouvelle classe et d'un nouveau datatype.
- Ajoutez un <classes> contenant un <memberOf> à la spécification de l'élément <book> pour qu'il participe à la nouvelle classe bookAtts
- Générez un schéma et assurez-vous que le fichier oddex-1-test.xml reste valide avec cette version du schéma.
- Contrôlez que oXygen vous propose ces nouveaux attributs, et qu'il contraint correctement les valeurs possibles

10. Qu'est-ce que l'on pourrait vouloir ajouter pour bien documenter son système ?

Peut-être…

- Des gloses, des descriptions en plusieurs langues
- Des exemples d'usage
- – modèles de contenu plus complexes

– contraintes variables selon le contexte

Et comme dans tout projet de documentation : indication de version, référencements extérieurs et intérieures, mappings ontologiques…

11. Exemples d'usage

Évidemment, si on décrit un schéma XML on va inclure des exemples d'usage en XML. Si la documentation s'exprime également en XML, il faut être astucieux… Il y a trois approches possibles :

- tout cacher avec un "CDATA marked section" (magique hérité de SGML)

```
<![CDATA[<p>un paragraphe</p>
              ]]>
```

- tout échapper en utilisent des "références" (`&lt;` etc)
- utiliser un autre espace de nommage

```
<egXML xmlns="http://www.tei-c.org/ns/examples"> <p>un paragraphe</p> </egXML>
```

Seul le dernier vous permet de valider vos exemples : un plus très avantageux

12. Tester votre compréhension...

- Revenez au fichier oddex-2.odd
- Modifiez la définition de l'attribut *@status* pour lui permettre de prendre la valeur `orange`.
- Ajoutez des traductions des descriptions d'éléments dans d'autres langues (allemand, russe, italien, anglais...)
- Modifiez l'élément `<para>` pour lui donner les attributs *@xml:id* et *@status*
- Ajoutez un nouvel élément `<chapitre>`, participant à la classe bookPart et contenant des membres de la meme classe. Quel est son effet?
- Ajoutez un exemple d'usage pour quelques-uns des éléments spécifiés par votre ODD

13. Et finalement un mot de la TEI

Admettons enfin que notre `<para>` n'est pas si loin de l'élément TEI `<p>`, que notre `<image>` ressemble beaucoup à l'élément TEI `<graphic>`, et que notre `<book>` pourrait être considéré comme un élément TEI `<div>`. Comment ré-écrire ce schéma pour profiter des définitions TEI existantes ?

```
<schemaSpec start="div"
 ident="testSchema-2" source="tei:1.6.0">
 <elementRef key="div"/>
 <elementRef key="p"/>
 <elementRef key="graphic"/>
 <elementRef key="figure"/>
 <moduleRef key="tei"/>
</schemaSpec>
```

Le module tei proposé par ce `<moduleRef>` nous fournit un ensemble de définitions infrastructurales, notamment pour les classes utilisées partout dans le systeme TEI. À part cela, nous n'avons besoin que de référencer les éléments TEI souhaités avec un `<elementRef>`.

L'attribut *@source* indique la version de la TEI que nous souhaitons utiliser.

- Chargez le fichier oddex-tei.odd avec oXygen et comparez le avec les versions précédentes
- Transformez ce fichier en schéma, comme d'habitude.
- Le fichier oddex-tei-test.xml contient une version TEI de notre fichier de test initial : validez-le avec le schéma que vous venez de créer.
- Notez qu'un document TEI *doit* utiliser l'espace de nommage TEI
- Notez également que les concepts TEI et les nôtres ne sont pas forcément identiques (par ex, usage de `<graphic>`)

14. Deuxième exercice un peu plus vraisemblable

Pour notre projet de corpus langagier, 'TEI Lite' est trop lourd, et 'TEI simplePrint' trop complexe. Créons un schéma TEI hyperLeger très simple.

- L'entête TEI ne possédera que les composants essentiels à sa conformité (à savoir : `<teiHeader>`, `<fileDesc>`, `<titleStmt>`, `<publicationStmt>` `<sourceDesc>` et `<title>`)
- L'élément `<text>` sera composé d'un seul `<body>` contenant des `<div>`s contenant un ou plus d'éléments `<ab>`
- Chaque `<ab>` ne permettra qu'une séquence d'éléments `<s>`
- Chaque `<s>` est composé d'éléments `<w>`
- Nous souhaitons supprimer la majorité des attributs, en gardant notamment *`@xml:id`*, *`@xml:lang`*, *`@type`*, *`@rend`*
- Ouvrez le fichier superLite-1.odd avec oXygen
- Testez votre compréhension de ce fichier
- Utilisez oXygen pour générer un schéma dans votre langage de schéma préféré et un petit manuel en HTML
- Créez un nouveau document TEI XML qui utilise ce schéma
- Vérifiez les éléments et les attributs disponibles

Nous allons modifier un peu plus ce schéma pour :

1. supprimer plusieurs attributs inutiles
2. ajouter un nouveau élément
3. simplifier le modèle de contenu de quelques éléments

15. (1) Suppression d'attributs

Pour supprimer un attribut il faut savoir d'où il vient :

- soit il est défini avec l'élément qui le comporte (assez inhabituel)
- soit il est hérité d'une classe
- qui peut elle-même hériter des attributs d'une autre classe

On peut supprimer ou modifier un attribut

- en modifiant que l'élement qui le comporte (effet local)
- en modifiant la participation à la classe qui le fournit (effet local)
- en modifiant ou supprimant la classe entièrement (effet global)

La participation d'un élément aux classes est explicitée dans sa définition. Par exemple:

Attributes att.global (@xml:id, @n, @xml:lang, @xml:base, @xml:space) (att.global.rendition (@rend, @style, @rendition)) (att.global.responsibility (@cert, @resp)) (att.global.source (@source)) att.declaring (@decls)

Pour savoir quels autres éléments participent à une classe, on peut regarder la définition de la classe elle même

16. Exemples

Modifiez le fichier superLite-1.odd :

- supprimez partout les attributs *`@decls`* et *`@source`*:

```
<classSpec type="atts" mode="delete"
 ident="att.declaring"/>
<classSpec type="atts" mode="delete"
 ident="att.source"/>
```

- la classe att.global.rendition nous fournit 3 attributs, et nous ne souhaitons que *`@rend`*: il y a 2 manières de supprimer les autres :

```
<classRef type="att.global.rendition"
 except="style rendition"/>
<!-- ou comme ceci: -->
<classRef type="att.global.rendition"
 include="rend"/>
```

(quelle est la différence?)

- supprimez partout les attributs *@xml:base* et *@xml:space*

```
<classSpec type="atts" ident="att.global"
 mode="change">
 <attList>
  <attDef ident="xml:base" mode="delete"/>
  <attDef ident="xml:space" mode="delete"/>
 </attList>
</classSpec>
```

- supprimez partout les attributs *@cert* et *@resp* : ces attributs sont fournis par la classe att.global.responsibility à laquelle participe att.global, donc nous modifions la liste des sur-classes pour att.global:

```
<classes>
 <memberOf key="att.global.rendition"/>
</classes>
```

Dans ce dernier cas, l'élément <classes> est forcément à remplacer entièrement, puisque ses enfants <memberOf> ne sont pas identifiables, et donc ne peuvent pas être ciblés par *@key*

Effectuez ces modifications dans le fichier superLite-1.odd et l'enregistrez sous le nom de superLite-2.odd si tout se passe bien...

17. (2) Ajout d'un nouvel élément

Nous souhaitons ajouter un élément <mw> pour baliser les 'multiwords' (concept linguistique un peu inexacte mais...)

- il faut fournir un <elementSpec> complet
- le nouvel élément devrait être dans notre propre espace de noms
- il est avantageux de le faire participer à une des classes TEI existantes pour qu'il apparaissent dans les modèles de contenu relevants

```
<elementSpec ident="mw"
 ns="http://myNameSpace.org">
 <desc>multiword strings which function as if they were single words</desc>
 <classes>
  <memberOf key="model.segLike"/>
  <memberOf key="att.global"/>
  <memberOf key="att.typed"/>
 </classes>
 <content>
  <alternate maxOccurs="unbounded">
   <textNode/>
   <elementRef key="w"/>
  </alternate>
 </content>
</elementSpec>
```

Modifiez ainsi votre fichier superLite-2.odd et voir l'effet sur le schéma généré et sa documentation.

18. (3) Modification du modèle de contenu

- La plupart des modèles de contenu s'exprime avec un macro, qui fait référence à des classes
- Regardez par ex la spécification de l'élément <s> dans notre petit schéma
- Le macro macro.phraseSeq définit un modèle permettant de texte ou des éléments des classes model.gLike, model.phrase et model.global
- Suivez l'hiérarchie des déclarations et vous constaterez que la plupart de ces classes sont vides dans notre petit schéma
- En effet, les seuls éléments disponibles sont <title>, <s>, <w> et (après la modification précédente) <mw>

Nous souhaitons simplifier cela : d'abord nous allons changer le <content> de <s>.

19. Changement du modèle de contenu

Ajouter dans votre ODD :

```
<elementSpec ident="s" mode="change">
```

```
 <content>
  <alternate maxOccurs="unbounded">
   <elementRef key="w"/>
   <elementRef key="mw"/>
  </alternate>
 </content>
</elementSpec>
```

- Regénérez le schéma et la doc, et contrôlez si l'effet corresponde à vos attentes
- Vous avez noté sans doute que l'exemple d'usage dans la doc n'est plus valide! il faut donc le remplacer. Ajoutez donc un element <exemplum> apres le <content>, qui contient un egXML xmlns="http://www.tei-c.org/ns/Examples" dont le contenu pourrait etre (par ex)

```
<s>
 <w>Quand</w>
 <w>partez</w>
 <w>-</w>
 <w>vous</w>
 <w> ?</w>
</s>
<s>
 <w>Demain</w>
 <w>.</w>
</s>
```

- Regénérez votre schéma et sa documentation; controllez la doc pour voir si votre exemple y apparait.
- Comme les <memberOf>, les <egXML> ne sont pas identifiables; ajouter un sert à remplacer tous

Attention : un changement de modèle de contenu risque de mettre en cause notre TEI-conformité

20. Une autre methode pour la modification des modeles

Comment modifier notre ODD pour supprimer <title>, sauf dans le <titleStmt> où il est obligatoire ?
Rappelons qu'un element <x> devient disponible dans le contenu d'un autre element <y> par deux moyens:

- <x> est referencie explicitement par le modele de contenu de <y>
- <x> est membre d'une classe qui est referenciee par le modele de contenu de <y>
- ou d'une classe qui est referencie par un macro definissant le modele de contenu de <y>

Regardez dans la doc ...

- comparez les modeles de contenu de <titleStmt> et de <ab>
- le contenu de <ab> est defini par un macro : suivez l'hierarchie de classes pour retrouver celle a laquelle appartient <title>!
- comment changer l'adhesion de <title> a cette classe?

21. Exercice final

S'il vous reste du temps, je vous invite à ....

- ajouter un element <pc> pour les séquences de ponctuation
- ajouter encore des exemples d'usage partout
- supprimer l'attribut *@hand* sur <ab> (ou partout)
- prédéfinir une liste fermée de valeurs possible pour l'attribut *@type* de l'élément <ab>

22. Pour aller plus loin

- tutoriel pratique sur l'enchainement des ODD
-
- tutoriel pratique sur la generation des ODD
- tutoriel pure ODD (en anglais)
- le manuel: TEI Guidelines, chap 22
- tutoriel introductif sur usage de Roma pour creer un ODD
- n'hesitez pas de posez vos questions sur TEI-L ou tei-fr